## REMARKS

This responds to the Office Action mailed on September 8, 2004. No claims are canceled, amended, or added herein; as a result, claims 1-31 remain pending in this Application.

### §103 Rejections of the Claims under Dundas and Ukai

Claims 1, 12, and 14-29 were rejected under 35 USC § 103(a) as being unpatentable over Dundas et al., "Improving Data Cache Performance by Pre-executing Instructions Under a Cache Miss," in view of Ukai et al. (U.S. 5,983,324). Applicant respectfully traverses these rejections.

As noted in a previous paper, the Dundas paper shows a run-ahead system that has no suggestion of any way to protect cache lines against eviction. In fact, Applicant finds no mention in Dundas that cache lines are evicted at all during run-ahead mode. He states only that cache misses are placed in a queue. Further, even if Dundas did evict lines without protection in run-ahead mode, his system would not fail, but would only run slower. For these reasons, the Dundas reference itself not only does not recognize an overflow problem, but might not have the problem in the first place. Under either of these conditions, Dundas provides no motivation for adducing any other reference related to this problem.

The Office Action in par. 8 (pages 9-10) states that

> "Applicant's argument is based on assuming that the invention of Dundas does not intend to execute cache misses during run-ahead when it places those missed in a queue. This is incorrect. Placing such misses in a queue does not imply that such misses will only be dealt with after run-ahead, the system can in fact be dealing with the misses in the queue's order while run-ahead is taking place. In such a case, protection of the pre-executed data is necessary and thus, a need for the modification of the system to include the limitations of Ukai can be understood."

Let us turn this argument around.

> The Examiner's argument is based upon assuming that the disclosure of Dundas *does* intend to execute cache misses during run-ahead when it places those cache misses in a queue. This is incorrect. Placing such misses in a queue does not imply that such misses will be dealt with *during* run-ahead; the system can in fact merely store them and terminate run-ahead.[1] In such a case, protection of the pre-executed instructions is *unnecessary*, and *no* need for modification of the system to include the disclosure of Ukai can be understood.

[1] --Termination could easily require a large number of cache cycles, so that multiple misses might possibly occur in the queue.

That is, the Office Action assumes that Dundas operates in a certain way, when in fact Dundas simply does not say one way or the other. And the Examiner's assumption is grounded in Applicant's disclosure. This is what is impermissible under 35 USC §103. *In re Mills*[2] states that the fact that certain references *can* be combined does not render the combination obvious, unless the prior art also suggests the *desirability* of the combination. *In re Sang Su Lee*[3] requires specific objective evidence in the references themselves for motivation to combine.[4]

As to Ukai, par 10, page 6 of the Office Action declares that "Dundas does not teach cache line protection; this limitation being taught by Ukai." But Ukai concerns a buffer for data, and not a "cache" for "instructions." A buffer need operate only at the peak speed of its input data stream, and each data item is pulled from the buffer once by the output stream. A cache, on the other hand, is practical only if it operates at many times the speed of its input memory, because a cache bets its entire usefulness upon multiple accesses of the same cache items by the output stream. These are two different animals. The only reason a buffer requires any protection is that its source stream may become faster than the output stream long enough to overfill the buffer. This condition can never happen in an instruction cache that has no run-ahead mode. Because Dundas' disclosure never informs the art that this problem may arise in a run-ahead instruction cache---and because correct operation of his system did not require its solution even if he might have in his own mind recognized its existence---this provides no motivation for an artisan reading the Dundas reference to combine it with Ukai..

Par. 10 of the Office Action then states that "Ukai does not teach a normal mode during which instructions are executed directly from a cache, and a run-ahead mode where future instructions are being executed; threes limitations are taught by Dundas." In fact, Ukai stores only file data requested by application programs, and does not store "instructions" at all..

---

[2] --16 USPQ2d 1430 (Fed. Cir. 1990)

[3] -- 61 USPQ2d 1430 (Fed. Cir. 2002)

[4] --These two decisions expand upon the older CCPA decisions of *In re Nomiya*, 184 USPQ 607 (CCPA 1975), *In re Bozek*, 163 USPQ 545 (CCPA 1969), and In re McLaughlin, 170 USPQ 209 (CCPA 1971), cited in the Office Action. In particular, the Office Action paraphrases In re McLaughlin to the effect that

> "[I]t must be recognized that any judgment on obviousness is based upon hindsight reasoning. But so long as it takes into account only knowledge which was within the level of ordinary skill at the time the invention was made, and does not include knowledge gleaned *only* from the applicant's disclosure, such a reconstruction is proper." (Office Action, par. 9, page 9; emphasis supplied)

This is not now a sufficient basis for a rejection under 35 USC §103, and probably never was.

Further, has no need of multiple modes for any discernible purpose. Therefore, this statement in the Office Action provides no motivation in Ukai to combine it with Dundas.

We are thus left with a combination whose only motivation in the primary reference is the solution of a problem whose existence and recognition in that reference are assumed without objective evidence, and whose solution from the secondary reference requires wholesale modification to introduce items (the modes) that would have no purpose therein.

For these and other reasons, the "instruction cache," the "normal mode" and the "run-ahead mode" of claim 1 distinguish it in a patentable manner from any permissible combination of the references.

Article claim 29 follows the method recitations of claim 1, and thus distinguishes the references for the same reasons.

Independent claim 12 sets and clears "protection bits" which find no resonance in Dundas, because Dundas suggests no protection of any kind, nor any need for protection. Conversely, Ukai has neither normal threads that "execute only valid instructions" nor a prefetching thread that "executes only future instructions that do not produce valid results." Ukai stores only data, and not instructions. None of Ukai's data produces "invalid results" when accessed by its requesting program. Further, in Ukai's system each requesting program accesses data only from its own thread; Ukai has no suggestion of allowing data allocated for one program to be accessed by a different program, as in claim 12's recitation for clearing protection bits when threads "allocated for the ... prefetching thread are referenced by the normal threads." That is, in addition to the impermissibility of the combination, even an improper combination does not reach claim 12.

Claims 13, 14, and 15 depend from claim 12, and distinguish the references for at least the same reasons. In addition, claim 14 spawns a prefetching thread "for a particular section of code." This finds no resonance in the cited references. The Office Action rejection on page 3 that "prefetching algorithms are implemented reliably in software" is irrelevant to the claim recitations.

Independent apparatus claim 16 stores data "for instructions to be executed by [a] processor," unlike Ukai's file data requested by programs. Also finding no suggestion in Ukai are the "normal" and "prefetch" modes, as discussed previously. Then claim 16 recites

identifiers "to indicate whether to protect an associated cache line." Ukai does not show a "cache," and Dundas suggests no protection. The statements on page 4 of the Office Action for this claim parallel those on pages 9-10 of the Action, and are equally invalid: "**eviction becomes necessary in the system of Dundas et al.**" (emphasis in original) might (or might not) be true, but it does not specify where and when the eviction takes place; Dundas does not tell us under what circumstances eviction might take place, or whether a putative eviction method might benefit from protection.[5]

Claims 17-20 depend from claim 16, and import all its recitations, and annex other features as well. For example, claim 20 employs a "cache controller" for storing the protection identifiers for the cache. No corresponding item is found in the references, nor does the Office Action assert that these two references by themselves show a controller.

Independent claim 25, like claim 16, recites two different modes, not found in Ukai for any purpose. Claim 25 then recites the "protection bit" not found in Dundas. As before, the rejection of claim 25 contravenes 35 USC §103 by forcing Dundas and Ukai into an unnatural alliance that neither would have contemplated in the absence of Applicant's disclosure.

Dependent claims 26-28 incorporate the recitations of their parent.

Independent claim 21 also recites the "run ahead mode" and "normal mode" for which Ukai has neither suggestion nor purpose, and includes the valid and invalid instructions that are so foreign to Ukai. The "protection bit" associated with the cache lines is again neither required nor suggested by Dundas.

In addition to the differences mentioned earlier, claim 21 recites "multiple processors" Dundas has only a single processor. Ukai has multiple programs, but---as far as Ukai tells us---has only a single processor. However, even if Ukai's application programs could be shoehorned onto multiple processors, the whole point of Ukai's system is that a single buffer pool serves all the programs, whereas claim 21 specifies "a plurality of caches each one of the caches associated with one of the processors." This is the antithesis of Ukai's teaching on how to combine multiple buffers into a single entity to allow one program to take more than its aliquot share of

[5] -- Again, Dundas does not require protection. His system will operate correctly without any protection. It might merely operate more slowly than one that incorporates the present invention.

the total buffer space. Thus, even an improper combination of the references fails to reach claim 21.

Claims 22-24 depend from claim 21, and bring other features as well. For example, claim 24 recites a "plurality" of tag arrays, whereas Ukai has only one (and Dundas has none).

## §103 Rejections of the Claims under Dundas, Ukai, and Microsoft

Claims 20, 22, and 23 were rejected under 35 USC § 103(a) as being unpatentable over Dundas et al. in view of Ukai et al. and Microsoft Computer Dictionary. These rejections are respectfully traversed. These claims were also rejected over Dundas and Ukai, without the tertiary reference to Microsoft. The discussion above demonstrates that their parent claims, 16 and 21, distinguish over these references. The Microsoft dictionary merely adds a definition of a generic "controller" as a device employed by other devices for access to a computer subsystem. This broad definition adds nothing to the parent-claim elements, so that the dependent claims 20, 22, and 23, which incorporate all of their elements, still distinguish over the any proper combination of the three references.

## §103 Rejections of the Claims under Dundas, Ukai, and Petrick

Claims 2-3, 6-7, 9-10, and 30-31 were rejected under 35 USC § 103(a) as being unpatentable over Dundas et al. in view of Ukai et al., further in view of Petrick et al. (U.S. 5,920,889).

Claims 2-3 depend from claim 1 and inherit all its recitations. In attempting to rebut Applicant's previous argument that adding Petrick to Dundas and Ukai creates the problem that Applicant has solved, the Office Action declares:

> "This statement assumes that the invention of Dundas does not intend to execute cache misses during run-ahead when it places those missed in a queue. This is incorrect. Placing such misses in a queue does not imply that such misses will be dealt with after run-ahead, the system can in fact be dealing with the misses in the queue's order while run-ahead is taking place." (par. 9, page 10)

As in the case of claim 1, this is pure supposition. Applicant finds no suggestion in Dundas that this is actually what happens. The opposite supposition may be made just as easily, viz:

> This statement assumes that the invention of Dundas does not intend to execute cache misses during run-ahead when it places those missed in a queue. The reference is silent on this point. Placing such misses in a queue does not imply that such misses will be

> dealt with *during* run-ahead as in Applicant's system, the system can in fact be dealing with the misses in the queue's order *after* run-ahead has completed.

Therefore, the subsequent statement in the Office action does not logically follow.

> In such a case protection of the pre-executed data is *necessary* and thus need for modification of the system to include the limitations of Ukai can be *understood*. This in it of it's self [*sic*] hints at the need to evict data to make room for the prefetches that arise from the cache misses during run-ahead execution; thus the teachings of Petrick et al. would be beneficial to the system of Dundas in view of Ukai." (par. 9, page 10; emphasis supplied)

In the first place, as shown above, Dundas does not state that he evicts cache lines during prefetch. Because there is no inherent or logical reason that eviction *necessarily* takes place during the prefetch operation itself, one cannot merely read into a reference an assumption as to how it *could* work---especially when that assumption springs from Applicant's disclosure. Secondly, even if Dundas might evict cache lines during prefetch, protection is still not "necessary." As noted above and previously, evicting an unused cache line does not cause the system to fail. At worst, the full benefit of some of the prefetching might be degraded. Tests might show that in some systems this degradation might not be worth the cost of extra facilities to protect against it. Therefore, the addition of Petrick to evict cache lines finds no motivation in Dundas, except through Applicant's teaching.[6]

Article claims 30-31 include the method recitations of claims 2-3 respectively, and thus distinguish the prior art for the same reasons.

Independent claim 6 includes the separate "normal mode" and "run ahead mode" of previously discussed claims, and further includes their "protection bit." The combination of Dundas with Ukai is impermissible for the same reasons as discussed in connection with claims such as method claims 1 and 12. The addition of Petrick does not cure any of those deficiencies; it has no run-ahead mode, for example, and does not solve any problem that is taught or suggested in Dundas. Therefore, no motivation exists for adding anything such as Petrick. Petrick stands on the same footing as Ukai in this regard; Dundas either does not evict cache lines during run-ahead, or chooses to disregard any performance hit that premature eviction might---or might not---entail.

Claim 7 depends from claim 6, and distinguishes the prior art for the same reasons.

---

[6] -- Nor does Ukai provide motivation. Ukai already includes protection of his data buffer; it is difficult to see how Petrick might add anything at all to Ukai.

Independent claim 9 also contains modes for "run-ahead execution" and "normal execution," not found in Ukai or Petrick Claim 7 further recites the "protection bit" for cache-line replacement that is not found in Dundas. The combination of these three references is an impermissible hindsight reconstruction for the reasons given above.

Claim 10 depends from claim 9, and inherits its elements.

## Allowable Subject Matter

Claims 4, 5, 8, 11, and 13 were objected to as being dependent upon a rejected base claim, but were indicated to be allowable if rewritten in independent form including all of the limitations of the base claim and any intervening claims. These claims are not rewritten herein, pending the resolution of the rejections of their parent claims.

## Conclusion

For the above and other reasons, Applicant urges that all the claims presently in the subject Application define over the cited prior art in a patentable manner, and respectfully requests reexamination under 35 USC §132 and allowance thereof. The Examiner is invited to telephone Applicant's attorney at (612) 373-6971 to facilitate prosecution of this Application.

If necessary, please charge any additional fees or credit overpayment to Deposit Account No. 19-0743.

Respectfully submitted,

CHRISTOPHER B. WILKERSON

By his Representatives,

SCHWEGMAN, LUNDBERG, WOESSNER & KLUTH, P.A.
Attorneys for Intel Corporation
P.O. Box 2938
Minneapolis, Minnesota 55402
(612) 373-6971

Date 8 Dec. 2004 By _____
J. Michael Anglin
Reg. No. 24,916

CERTIFICATE UNDER 37 CFR 1.8: The undersigned hereby certifies that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail, in an envelope addressed to: MS Amendment, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on this 8th day of December, 2004.

Amy Moriarty
Name

_____
Signature